STANFORD UNIVERSITY, STANFORD, CALIFORNIA 94305

ROBERT H. CANNON, JR.
CHARLES LEE POWELL PROFESSOR
AND CHAIRMAN
DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS

October 3, 1989

Dr. Henry Lum, Jr.
NASA-Ames Research Center
Information Sciences Office, 244-7
Moffett Field, CA 94035

Dear Dr. Lum:

Enclosed are two copies of the the 8th Semi-Annual Report on Research on Control of Free-Flying Space Robot Manipulator Systems for the period September 1988 through February 1989.

Sincerely,

[illegible]

Encls

EIGHTH SEMI-ANNUAL REPORT

ON

RESEARCH ON

CONTROL OF FREE-FLYING SPACE ROBOT MANIPULATOR SYSTEMS

Submitted to

Dr. Henry Lum, Jr., Chief, Information Sciences Division
Ames Research Center, Moffett Field, CA 94035

by

The Stanford University Aerospace Robotics Laboratory
Department of Aeronautics and Astronautics
Stanford University, Stanford, CA 94305

Research Performed Under NASA Contract NCC 2-333
During the period September 1988 through February 1989

Professor Robert H. Cannon Jr.
Principal Investigator

$\sim$

Contents

List of Tables

$\sim$

List of Figures

∼

Chapter 1

Introduction

This document reports on the work done under NASA Cooperative Agreement NCC 2-333 during the period August 1988 through February 1989. The research was carried out by a team of six Ph.D. candidate students from the Stanford University Aerospace Robotics Laboratory under the direction of Professor Robert H. Cannon, Jr. The goal of this research is to develop and test experimentally new control techniques for self-contained, autonomous free-flying space robots. Free-flying space robots are envisioned as a key element of any successful long term presence in space. These robots must be capable of performing the assembly, maintenance, and inspection, and repair tasks that currently require astronaut extra-vehicular activity (EVA). Use of robots will provide economic savings as well as improved astronaut safety by reducing and in many cased eliminating the need for human EVA.

The focus of our work is to develop and carry out a set of research projects using laboratory models of satellite robots. These devices use air-cushion-vehicle (ACV) technology to simulate in two dimensions the drag-free, zero-g conditions of space. Using two large granite surface plates (6' by 12' and 9' by 12') which serve as the platforms for these experiments we are able to reduce gravity-induced accelerations to under $10^{-5}g$ with a corresponding drag-to-weight ratio of about 10^{-4}—a very good approximation to the actual conditions in space.

Our current work is divided into six major research projects: Cooperative Manipulation from a fixed-base robot, Cooperative Manipulation from a Free-Flying Robot, Global Navigation and Control of a Free-Floating Robot, Multiple-Robot Cooperation, an alternative transport mode called LEAP (Locomotion Enhancement via Arm Push-Off), and Adaptive Control of LEAP. All experiments done to verify functionality and measure performance of developed theory will be performed on our second generation space-robot model which is operational, but does not yet have global and endpoint sensing.

The Fixed-Base Cooperative Manipulation experiment represents a major breakthrough in robotics: object level control of a cooperative two-arm manipulation system. The human operator interacts with the system via a computer graphical interface, allowing him to designate target objects to catch and manipulate objects. Operator actions are given to the system as high-level input. The cooperating arms will then be managed by the system to carry out the given instructions. Experiments have successfully demonstrated throwing and catching a small air cushion supported vehicle moving in two dimensions, and inserting either of its end fixtures into any of several mating parts.

The Global Control and Navigation project seeks to demonstrate simultaneous control of the robot manipulators and the robot base position on the free-flying robot model. This will allow manipulation tasks to be accomplished while the robot body is controlled along a trajectory.

The Free-Flying Cooperative Manipulation project has demonstrated, in preliminary experiments, the ability to manipulate an object using cooperative arms from a free-flying robot. In the course of this research, catching, manipulating, and throwing a free-flying object with a free-flying robot will be demonstrated.

The LEAP project seeks to demonstrate locomotion of a free-flying robot via the use of its manipulators. This will provide a viable alternative to expendable gas thrusters for vehicle propulsion. This work will be carried out with a slightly revised version of the second generation space robot model.

The Multiple-Vehicle Cooperation project will demonstrate multiple free-floating robots working in teams to carry out tasks too difficult or complex for a single robot to perform. A third space robot model is also currently under construction; and it will bring the fleet of free-flying robot models to three.

The Adaptive LEAP project seeks to bring the benefits of adaptive control technology to free-flying robots. The successful execution of the LEAP technique and precise manipulation of unknown payloads requires an accurate model of the robot and payload mass properties. The properties of free-flying robots make them an attractive testbed for advanced adaptive control technology.

The chapters that follow give detailed progress and status reports on a project-by-project basis.

Chapter 2

Experiments in Fixed-Base Cooperative Manipulation

Stan Schneider

2.1 Introduction

To accelerate our development of multi-armed, free-flying satellite manipulators, we have developed a fixed-base cooperative manipulation facility. Although the manipulator arms are fixed to a rigid base, they manipulate free-flying objects. This facility allows us to experiment quickly with cooperative algorithms, expediting our study of space-based manipulation and assembly.

2.1.1 Research Goals

The goal of this project is to study *simultaneously* the dynamic and strategic issues of robotic manipulation, and to demonstrate experimentally multi-arm cooperative robotic object acquisition and assembly. The aim is not only to master the dynamic control problem, but also to provide simple, conceptual *direction* of motion by an untrained operator. This goal forces treatment of the system as a whole. By focusing directly on the vertical integration problem, we are studying not only the various subsystems, but also their interfaces and interactions.

More specifically, the goals of this project are to:

- Develop a complete cooperative robotic system hierarchy.
- Provide simple, conceptual direction of motion.
- Implement dynamically-compensated multiple-arm control.
- Integrate real-time visual feedback.
- Verify the design experimentally.

2.2 Status

This initial research effort into cooperative manipulation was completed during the previous report period; much of the technical detail is contained in our last report [4]. Two published papers [12, 13], attached as appendix A, describe in full early experimental results. A PhD thesis [11] is nearing completion, and will be attached to the next report.

In summary, high-performance cooperative dynamic control has been demonstrated, both in "free" motion and when the manipulated object is in contact with its environment. The point-tracking vision system is capable of identifying and tracking multiple moving objects. Each object's position and orientation (in the plane) is determined in real-time with sub-pixel resolution. Autonomous dual-arm capture, docking (connector insertion), withdrawal, and throwing functions are supported by the strategic command module.

The mouse-based graphical user interface allows an operator to direct the activities of the system at the conceptual level. The operator commands only object motions; the arm actions required to effect these motions need not be specified. This design allows simple specification of many tasks. In particular, simple assembly sequences can be easily accomplished. Each of these functions has been fully demonstrated experimentally.

2.3 Progress Summary

During this report period, activity focused on technology transfer and documentation.

Much of the software developed for this project was ported to the VxWorks environment, and thus to the free-floating robot platform. In particular, the Controlshell real-time environment and Scope data collection and display utility are operational under VxWorks. Several simple controllers have been demonstrated on the free-flying robot utilizing these tools.

Chapter 3

Experiments in Navigation and Control of Free-Flying Space Robots

Marc Ullman

3.1 Introduction

This chapter summarizes the progress to date in our research on global navigation and control of free-flying space robots. This work represents one of the key aspects of our comprehensive approach to developing new technology for space automation. Ultimately, we envision groups of fully-self contained mobile robots making up the core work force in space.

3.1.1 Motivation

Although space presents us with an exciting new frontier for science and manufacturing, it has proven to be a costly and dangerous place for people. Space is therefore an ideal environment for sophisticated robots capable of performing tasks that currently require the active participation of astronauts.

While earth based robots have not always proved to be cost effective solutions to manufacturing inefficiencies (due to the abundance of cheap labor), the tremendous cost associated with putting humans in space, especially when EVA is required, makes the economics of robots in space particularly attractive.

3.1.2 Research Goals

The immediate goals of this project are to:

- demonstrate the ability to simultaneously control robot base position and arm orientation so that a free-flying robot can navigate to a specified location in space while manipulating its arms.

- demonstrate the ability to capture a (possibly moving) free-floating target "on-the-fly" using the manipulator arms while the base is in transit.
- provide a suitable platform for the eventual addition of A.I. based path planning and obstacle avoidance algorithms which will enhance the robustness of task execution.

3.1.3 Background

This work emphasizes the modeling of robot dynamics and the development of new control strategies for dealing with problems of:

- a non-inertially fixed base (i.e. free-floating base)
- redundancy with dissimilar actuators
- combined linear and non-linear actuators
- highly non-linear dynamics
- unstructured environments

Our laboratory work involves the use of a model satellite robot which operates in two-dimensions using air-cushion technology. We have developed a series of satellite robots which, in two dimensions, experience the drag-free and zero-g characteristics of space. These robots are fully self-contained vehicles with onboard gas supplies, propulsion, electrical power, computers, and vision systems. The latest generation of robots is also equipped with a pair of two-link arms for acquiring and manipulating target objects.

3.2 Progress Summary

The following advances have been achieved during the past report period:

- The I/O transition module was designed, fabricated, and installed.
- Most of the wiring for the sensors and actuators is now complete.
- The plunger-type gripper end effectors were installed and plumbed.
- *Controlshell*and *Scope*have been ported to the VxWorks environment.
- Various bugs in VxWorks and dbxWorks have been identified and fixed.
- Rterm was developed to provide multi-window access to our real-time computers.
- A basic PD control loop was closed around the arms to verify the operation of both the robot hardware and the VxWorks realtime operating system with our enhancements.
- Work is progressing on our VME bus version of the Point Grabber Vision system.

3.3 Experimental Hardware

3.3.1 I/O Transition Module

The I/O Transition Module (IOTM) is a 6U high VME board which serves as a patch board/break-out box for all of the digital and analog signals that enter/leave the computer I/O system. The front panel contains two DB-25 and two DB-37 connectors which are routed to four sets of 3M solderless connectors. Four 64 pin DIN connectors on the board are similarly routed to four sets of 3M solderless connectors. The IOTM allows us to configure the outputs from any four VME boards which use the "a" and "c" rows of the P2 connector to our set of "D" connectors which connect via ribbon cables down to the analog card cage below. Thus by simply changing the patch board connections on the IOTM we can switch to different I/O boards without affecting the rest of our system.

With the IOTM complete, the majority of the sensor and actuator wiring is now complete.

3.3.2 Grippers

A pair of plunger style grippers identical to those used in our fixed-base cooperative control experiment has been added to the manipulator arms. These grippers allow a free-floating object to be grasped by inserting their Z-axis finger tip into a hole in the object. The Z-axis motion is produced by a double acting pneumatic cylinder which is controlled by a four-way solenoid valve.

3.4 Real-Time Computer System

3.4.1 *Controlshell*and *Scope*

*Controlshell*and *Scope*are two software tools that were developed during the course of our fixed-base cooperating arm work described in Chapter 2. Inasmuch as that work was done using the pSOS real-time kernel in a shared memory configuration, it was necessary to port these tools to run in the networked environment provided by VxWorks, our real-time operating system. *Controlshell*provides an object oriented framework for designing and testing real-time control software. It provides a set of objects for implementing interactive menu-driven user interfaces, dynamically loading and scheduling execution of control algorithms, and collecting and displaying real-time data. *Scope*is a companion to *Controlshell*that runs on a Sun Workstation and provides real-time oscilloscope style display of data from the real-time system. For a more detailed description of these tools, see reference [11].

3.4.2 VxWorks and dbxWorks

In the process of porting *Controlshell*and *Scope*to the VxWorks environment we uncovered several bugs in VxWorks. Through tenacious debugging we found and fixed many of them including a serious error in the memory management library. In the process of identifying and fixing these bugs, we developed several additional tools. One is a very complete heap-checking facility that can be enabled whenever one suspects that the heap is being corrupted.

dbxWorks is special version of the Sun UNIX source level debugger known as dbx. dbxWorks features a set of extensions to dbx that allows a Sun Workstation to be used for debugging an application that is actually running on a remote real-time system. This feat is accomplished by using a RPC (remote procedure call) based ptrace server that resides in the VxWorks kernel image. Although very useful in theory, our initial impression of this product was one of major disappointment; however, since Stanford has a source code license for the Sun operating system software, we were able to get the source to dbxWorks. We have since evolved it into a useful tool which comes much closer to meeting our original expectations. Among other things, we added a new feature that allows one to attach to (i.e. begin debugging) a process which is already running, a capability that was sorely missing in the original release.

3.4.3 *rterm*—a Multi-Window Remote Terminal Server

One of the more objectionable shortcomings of VxWorks that became immediately apparent during the above porting activities was the inability to support multiple logins to the target real-time system. This restriction meant that running an interactive application in the login window forced one to give up the "shell" or "console" connection since the application was now using the connection for its own I/O. This obstacle was overcome by building a new tool we call *rterm*. *rterm*is a multi-window remote terminal server that runs on a Sun Workstation. It accepts connections from client real-time systems and opens up a new virtual terminal window for each connection similar in concept to the networked client-server design of X-windows.[1] *rterm*allows applications running on the real-time system to open up new windows on the host Workstation from which the current login session was initiated and has proven to be quite useful. For example, *Controlshell*now automatically opens a new window when it is launched. In addition, a simple front end command called "window" now allows any VxWorks function to be run in its own window.

3.4.4 Operational PD Controller

A PD (proportional-derivative) controller was implemented using *Controlshell*that exercises all of the software tools described above. This controller verifies the proper operation of the manipulator hardware—sensors, actuators, electronics—as well as the computer system hardware and software. With our improvements, the VxWorks system is proving to be the useful and productive environment for building our control software that we had originally envisioned. In fact, we are planning on switching all of our experimental work (i.e. that work which is currently using the pSOS shared memory system) over to the VxWorks environment in yet another effort to help standardize our laboratory work so as to facilitate sharing of code and data.

[1]In fact we intend to do an X-windows version of *rterm*once we switch to XView—Sun's X-windows based windowing system. This version should be considerably simpler since most of the requisite functionality already exists within X itself.

3.4.5 Point Grabber Vision System

Work has been progressing on a VME bus version of our Point Grabber Vision system. This board takes the composite video signal from a NTSC CCD video camera and uses threshold detection logic to provide a list of pixels which exceed a preset threshold level. This list consisting of X and Y pixel values along with data valid bits for each camera (the board can handle up to four cameras simultaneously) is stored in a set of 512 x 9 FIFOS which can be read by the microprocessor upon receipt of an interrupt. The VME bus interface proved to be more challenging than originally anticipated; however, we believe that our prototype is now operating reliably.

The other difficulty we encountered involved trying to drive a pixel clock signal down a 15m transmission line, because we wanted the vision board and the cameras to be in sync. Although this is now working reasonably well, we are looking into using a new camera (the same model that was used in our fixed-base work) that is shutterable and has an internal phase-locked loop. This later feature should eliminate the need for sending the camera an external clock signal—we should be able to use the external horizontal sync signal instead. Since the vision system will be our primary position sensor for our global navigation work, it is imperative that we have a reliable and robust system.

3.5 Modeling and Simulation

3.6 Future Work

Our robot is ever closer to becoming operational. All major subsystem components are now in place. The principle work remaining entails some final wiring of auxiliary sensors and related equipment. Specific items yet to be completed include:

- Installing active target identification markings (LEDS) to the vehicle base, arm endpoints, and targets.
- Mounting the onboard and off-board cameras.
- Completing the vision system software and hardware to the point where it is operational and reliable.
- Mounting the angular rate sensor and the x-y accelerometers which will serve as the core of an onboard INS system for tracking vehicle position and orientation.

The modular design philosophy, which has been a guiding principle for this project since its inception, will continue to apply as our focus begins to shift away from hardware toward software.

$\sim$

Chapter 4

Experiments in Cooperative Manipulation from a Free-Flying Robot

Ross Koningstein

4.1 Introduction

This chapter summarizes the work performed on multiple arm cooperation on a free-flying robot under NASA grant NCC-2-333 for the period August 1988 to February 1989. Multi-arm cooperative manipulation is one of the basic technologies required for a space based robot. High-level user-interactive task specifications, as discussed in chapter 2, are desirable for commanding complex dynamical systems such as free-flying robots. In this experiment, the underlying technologies allowing cooperative manipulation from a free-flying base are being developed. The union of these two technologies will allow for graphically oriented high-level manipulation task specification.

Initial experiments have demonstrated that recursive algorithmic formulations for computed torque are suitable for implementation on real-time systems. This allows a simple, powerful solution of the computed torque [7] control problem for free-flying multi-armed systems possessing closed kinematic chains in some configurations. Using these algorithms, it is no longer necessary to hand-derive equations of motion for use in control systems - high performance can be achieved using preprogrammed algorithms acting on the robot's state.

4.2 Motivation

Most commercial robot control systems are based on joint by joint position and speed feedback (PD) control. While generally robust, PD controllers are not suitable for high-performance control of non-linear dynamical systems. To accurately compensate for time-varying geometries, one must consider the Computed Torque control method for such systems. A free-flying cooperating arm robot is a highly non-linear dynamical system, and requires a computed torque control method not only to compensate for time-varying geometries, but also to handle the peculiarities of a free-flying robot. In a free-flying robot, the robot arm endpoints ca be stationary in space while the arm's joints move due to base-body motion. The computed torque control scheme is not without its own problems, however: they rely on the inverse and derivative of the system's Jacobian, **J**.

The traditional Jacobian expresses manipulator endpoint speeds measured in a coordinate system as a function of the robot arm joint angles. In equation form this is expressed as

$$\mathbf{v}^{\text{endpoints}} = \mathbf{J}\dot{q}$$

where v is a vector of the speeds of the endpoints, and $\dot{q}$ are the derivatives of the system generalized coordinates. Free-flying robots have additional degrees of freedom: those of the robot body itself. Using the traditional Jacobian definition, it was not possible to formulate a square Jacobian for such systems, because the various manipulator endpoint positions have fewer degrees of freedom than the robot dynamical system. This problem has been addressed in two ways: by altering the method of solution for control torques, such as that done by Alexander[1], or by modifying the definition of the Jacobian to include additional system states [15]. The 'redundant' degrees of freedom of a free-flying robot have been analyzed in previous work; however, the case of dynamical constraints, such as those imposed by closed kinematic chains, have not been addressed. Traditionally, such dynamical systems with constraints are reduced in order for solution Kane[8]. This is not a desirable approach, since it involves the formulation and modification of all elements in the system inertia (mass) matrix. The algorithms presented to date in this research project do not generate an inertia matrix for reasons of efficiency. An alternate method of solving a constrained dynamical system for control purposes is discussed.

4.2.1 Concepts used in Analysis

This theory for serial chain manipulators is derived using Kane's dynamical analysis techniques. The analysis that follows assumes that the velocities v of points and angular velocities ω of bodies in the system under consideration can be expressed in a Newtonian reference frame as follows:

$$\mathbf{v}^i = \sum_{s=1}^{p} \mathbf{v}_s^i u_s$$

$$\omega^i = \sum_{s=1}^{p} \omega_s^i u_s$$

where the generalized speeds $u_{1..n}$ are linear combinations of the derivatives of the generalized coordinates $\dot{q}_{1..n}$. This will be true if no part of the system is undergoing prescribed motions. The partial angular velocities of bodies, and partial velocities of points, as defined by Kane[8], can be shown to be:

$$\begin{aligned} \mathbf{v}_r &= \frac{\partial}{\partial u_r}\mathbf{v} \\ \omega_r &= \frac{\partial}{\partial u_r}\omega \end{aligned}$$

4.3 The Dynamical System

The first step in controlling such a complex dynamical system is to identify the dynamical aspects of the system. The space robot being considered is a kinematic chain: each link connects serially to the next. We have derived a fast algorithm to formulate endpoint acceleration equations in kinematic chains[5]. This has been used to express desired endpoint accelerations for control purposes. The second aspect of this problem is the free-flying nature of the robot. The Jacobian can be augmented with momentum conservation equations[4]. The remaining problem is to deal with dynamical constraints introduced by the closed chain. This problem will be dealt with in the analysis that follows.

This analysis is performed for a free-flying robot torso with two kinematic chain manipulator arms which grasp a payload. The derivations are for 3D, but have been specialized to 2D for our experimental work. A generic free-flying robot has n degrees of freedom, which account for the degrees of freedom of the arms and the robot's torso, but subtracting those lost due to constraints such as chain closure or motion constraints (contact with other bodies). Notation used is that introduced by Kane[8].

4.4 Jacobian Structure

4.4.1 Desired Accelerations

First, the system Jacobian will be formulated using partial velocities. Then the desired endpoint accelerations will be expressed using these partial velocities and their derivatives. Desired accelerations are mapped through the system jacobian to determine joint accelerations, which is the basis for the computed torque method. The Jacobian, expressed using generalized speeds [1], is used as follows:

$$\mathbf{v}^{\text{endpoint}} = Ju$$

[1]If one chooses $u \triangleq \dot{q}$ then this is the standard Jacobian. If not, it becomes a more generalized Jacobian. The theory is valid for either case.

The endpoint acceleration can then be expressed as:

$$\mathbf{a}^{\text{endpoint}} = \mathbf{J}\dot{u} + \dot{\mathbf{J}}u$$

and the joint accelerations can be solved for by rearranging these equations:

$$\dot{u} = \mathbf{J}^{-1}(\mathbf{a}^{\text{endpoint}} - \dot{\mathbf{J}}\ u)$$

The Jacobian matrix's components are dependent upon the partial velocities and partial angular velocities of the endpoint of the manipulator(s) in the system. An endpoint's velocity can be expressed in terms of its partial velocities:

$$\mathbf{v}^{\text{endpoint}} = \sum_{r=1}^{n} \mathbf{v}_r^{\text{endpoint}} u_r$$

and therefore 3D endpoint velocity can be expressed in terms of speeds along some established inertial x,y and z directions, for example, along unit vectors which we define as **x**, **y** and **z**:

$$\begin{aligned}
\mathbf{v}^{\text{endpoint}} \cdot \hat{\mathbf{x}} &= \sum_{r=1}^{n} \mathbf{v}_r^{\text{endpoint}} \cdot \hat{\mathbf{x}}\ u_r \\
\mathbf{v}^{\text{endpoint}} \cdot \hat{\mathbf{y}} &= \sum_{r=1}^{n} \mathbf{v}_r^{\text{endpoint}} \cdot \hat{\mathbf{y}}\ u_r \\
\mathbf{v}^{\text{endpoint}} \cdot \hat{\mathbf{z}} &= \sum_{r=1}^{n} \mathbf{v}_r^{\text{endpoint}} \cdot \hat{\mathbf{z}}\ u_r
\end{aligned}$$

the elements of the Jacobian due to an endpoint's velocity is therefore:

$$\begin{aligned}
j_{1r} &= \mathbf{v}_r^{\text{endpoint}} \cdot \hat{\mathbf{x}} \\
j_{2r} &= \mathbf{v}_r^{\text{endpoint}} \cdot \hat{\mathbf{y}} \\
j_{3r} &= \mathbf{v}_r^{\text{endpoint}} \cdot \hat{\mathbf{z}} \\
&\vdots
\end{aligned}$$

As shown above, desired endpoint accelerations can be expressed in terms of the Jacobian, its derivative, and the generalized speeds and their derivatives. The derivatives of the elements of the Jacobian can also be determined from the partial velocities:

$$\begin{aligned}
\dot{j}_{1r} &= \dot{\mathbf{v}}_r^{\text{endpoint}} \cdot \hat{\mathbf{x}} \\
\dot{j}_{2r} &= \dot{\mathbf{v}}_r^{\text{endpoint}} \cdot \hat{\mathbf{y}} \\
\dot{j}_{3r} &= \dot{\mathbf{v}}_r^{\text{endpoint}} \cdot \hat{\mathbf{z}} \\
&\vdots
\end{aligned}$$

where the derivatives, taken in a Newtonian reference frame, of the partial velocities are

$$\dot{\mathbf{v}}_r^{\text{endpoint}} \triangleq \frac{{}^{N}d}{dt}\mathbf{v}_r^{\text{endpoint}}$$

These derivatives of partial velocities can calculated from partial velocities and the angular velocity of the body that the partial velocity vectors are based in. This completes the formal description of the Jacobian elements for desired accelerations. Note that desired angular accelerations are treated in an identical manner, allowing body angular acceleration specification.

4.4.2 Momentum Conservation

In any free-flying system of bodies, the linear and angular momenta vary according to the external forces and torques on the system. On a free-flying robot, these are the system thrusters. If assume that these thruster settings are known a priori, we are able to predict the rate of change of the system momenta. The Jacobian can be augmented with linear and angular momenta equations to include these system states in the calculation of the desired generalized accelerations. Inclusion of these relations can make a Jacobian full rank, and suitable for application of the computed torque method.

First, the linear momentum, then the angular momentum of the system will be examined. The linear momentum $\mathbf{L}^i$ of a body i in the system is

$$\begin{aligned}\mathbf{L}^i &= m^i\mathbf{v}^{i*}\\ &= m^i\sum_{s=1}^{n}\mathbf{v}_s^{i*}u_s\\ &= \sum_{s=1}^{n}\mathbf{L}_s^i u_s\end{aligned}$$

where the *partial linear momentum* of body i is defined by

$$\mathbf{L}_s^i \triangleq m^i\mathbf{v}_s^{i*}$$

The linear momentum $\mathbf{L}$ of a system of ν bodies is the sum of the linear momenta of each body i in the system:

$$\begin{aligned}\mathbf{L} &= \sum_{i=1}^{\nu}\mathbf{L}^i\\ &= \sum_{i=1}^{\nu}m^i\mathbf{v}^{i*}\\ &= \sum_{i=1}^{\nu}m^i\sum_{s=1}^{n}\mathbf{v}_s^{i*}u_s\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{\nu}\sum_{s=1}^{n} m^i \mathbf{v}_s^{i*} u_s \\
&= \sum_{i=1}^{\nu}\sum_{s=1}^{n} \mathbf{L}_s^i u_s \\
&= \sum_{s=1}^{n} \mathbf{L}_s u_s
\end{aligned}
$$

where the *partial linear momentum* of the system of ν bodies is defined by

$$\mathbf{L}_s \triangleq \sum_{i=1}^{\nu} m^i \mathbf{v}_s^{i*}$$

The *partial linear momenta* of the system can be formulated using the mass and center of mass partial velocity of each body in the system. The process of building an augmented Jacobian using these vector quantities is similar to the process used for the partial velocities discussed in the previous section, and will be examined after the angular momentum terms are derived.

The angular momentum $\mathbf{H}^i$ of each body i, about its center of mass is:

$$
\begin{aligned}
\mathbf{H}^i &= \mathbf{I}^{i/i*}\omega^i \\
&= \mathbf{I}^{i/i*}\sum_{s=1}^{n}\omega_s^i u_s \\
&= \sum_{s=1}^{n}\mathbf{I}^{i/i*}\omega_s^i u_s \\
&= \sum_{s=1}^{n}\mathbf{H}_s^i u_s
\end{aligned}
$$

where the *partial angular momentum* $\mathbf{H}_s^i$ of each body is defined as

$$\mathbf{H}_s^i \triangleq \mathbf{I}^{i/i*}\omega_s^i$$

The central angular momentum $\mathbf{H}$ of the system of ν bodies about the system's center of mass point, is:

$$
\begin{aligned}
\mathbf{H} &= \sum_{i=1}^{\nu}\mathbf{H}^i + \sum_{i=1}^{\nu}(\mathbf{r}^{i*} - \mathbf{r}^{cm}) \times m^i \mathbf{v}^{i*} \\
&= \sum_{i=1}^{\nu}(\mathbf{I}^{i/i*}\omega^i + (\mathbf{r}^{i*} - \mathbf{r}^{cm}) \times m^i \mathbf{v}^{i*}) \\
&= \sum_{i=1}^{\nu}(\sum_{s=1}^{n}\mathbf{I}^{i/i*}\omega_s^i u_s + \sum_{s=1}^{n}(\mathbf{r}^{i*} - \mathbf{r}^{cm}) \times m^i \mathbf{v}_s^{i*} u_s)
\end{aligned}
$$

$$= \sum_{i=1}^{\nu}\sum_{s=1}^{n}(\mathbf{H}_s^i u_s + (\mathbf{r}^{i*} - \mathbf{r}^{cm}) \times \mathbf{L}_s^i u_s)$$
$$= \sum_{s=1}^{n}\mathbf{H}_s u_s$$

where the *partial angular momentum* $\mathbf{H}_s$ of the system is defined as

$$\mathbf{H}_s \triangleq \sum_{i=1}^{\nu}(\mathbf{H}_s^i + (\mathbf{r}^{i*} - \mathbf{r}^{cm}) \times \mathbf{L}_s^i)$$

A set of Jacobian augmentation equations can be set up which describe the relation between the momenta and the generalized speeds.

$$\begin{aligned}\mathbf{L} &= \mathcal{J}^L u\\ \mathbf{H} &= \mathcal{J}^H u\end{aligned}$$

The elements of the Jacobian due to the linear and angular momenta are therefore:

$$\begin{aligned}\mathcal{J}^L_{1r} &= \mathbf{L}_r \cdot \hat{\mathbf{x}}\\ &\vdots\end{aligned}$$

and

$$\begin{aligned}\mathcal{J}^H_{1r} &= \mathbf{H}_r \cdot \hat{\mathbf{x}}\\ &\vdots\end{aligned}$$

The *partial momenta* can be formulated automatically using the partial velocities in the system.

Expected or desired momentum rates (due to external forces and torques) can be expressed in terms of these Jacobian augmentation equations and their derivatives along with the generalized speeds and their derivatives.

$$\begin{aligned}\dot{\mathbf{L}} &= \mathcal{J}^L \dot{u} + \dot{\mathcal{J}}^L u\\ \dot{\mathbf{H}} &= \mathcal{J}^H \dot{u} + \dot{\mathcal{J}}^H u\end{aligned}$$

The derivatives of the elements of the augmented Jacobian can be determined from the partial momenta:

$$\begin{aligned}\dot{\mathcal{J}}^L_{1r} &= \dot{\mathbf{L}}_r \cdot \hat{\mathbf{x}}\\ &\vdots\end{aligned}$$

and

$$\dot{j}_{1r}^{H} = \dot{\mathbf{H}}_r \cdot \hat{\mathbf{x}}$$
$$\vdots$$

where the derivatives, taken in a Newtonian reference frame, of the partial momenta are:

$$\dot{\mathbf{L}}_r \triangleq \frac{\mathrm{d}^N}{\mathrm{dt}}\mathbf{L}_r$$
$$\dot{\mathbf{H}}_r \triangleq \frac{\mathrm{d}^N}{\mathrm{dt}}\mathbf{H}_r$$

and the rate of change of the momenta are given by:

$$\dot{\mathbf{L}} = \sum \mathbf{F}^{\mathrm{ext}}$$
$$\dot{\mathbf{H}} = \sum \mathbf{T}^{\mathrm{ext}} + \sum (\mathbf{r}^{ext} - \mathbf{r}^*) \times \mathbf{F}^{\mathrm{ext}}$$

This completes the formal description of the Jacobian elements for momentum conservation.

4.4.3 Closed Chains

In a dynamical system with nonholonomic constraints, the generalized speeds $u_{1..n}$ are not independent. In a cooperating arm manipulator system this condition can arise when the arms form a closed chain. Such chain closure can introduce two types of velocity constraint. First, there will always be a velocity constraint which expresses the identical motion of both endpoints. Secondly, an angular velocity constraint is possible if the endpoints not only touch, but make a more rigid contact. The case of a velocity constraint introduced when performing cooperating arm manipulation will be analyzed, and the constraint equations will be expressed in terms of quantities used in the kinematics derivations.

The constraint of chain closure is described by:

$$\mathbf{V}^{\mathrm{endpoint}_1} = \mathbf{V}^{\mathrm{endpoint}_2}$$

expanding this into partial velocities,

$$\sum_{r=1}^{n} \mathbf{V}_r^{\mathrm{endpoint}_1} u_r = \sum_{r=1}^{n} \mathbf{V}_r^{\mathrm{endpoint}_2} u_r$$

defining a constraint velocity[2]:

[2]The concept of a constraint velocity is not dependent upon having a free-floating base and hence works for all instances of closed kinematic chains

$$\begin{aligned} \mathbf{C} &\triangleq \mathbf{v}^{\text{endpoint}_1} - \mathbf{v}^{\text{endpoint}_2} \\ &= 0 \end{aligned}$$

and the *constraint partial velocities*[3] evaluate to:

$$\mathbf{C}_r = \mathbf{v}_r^{\text{endpoint}_1} - \mathbf{v}_r^{\text{endpoint}_2}$$

It is evident that by dot multiplication with inertial basis vectors, as was done with endpoint velocity, this vector equation can be reduced to scalar equations for incorporation into the system Jacobian.

$$0 = \mathcal{J}^C u$$

where the elements of these Jacobian augmentation equations are:

$$\begin{aligned} \mathcal{J}_{1r}^C &= \mathbf{C}_r \cdot \hat{\mathbf{x}} \\ &\vdots \end{aligned}$$

These constraint partial velocities can be formulated automatically using the partial velocities of the endpoints of the manipulator which are touching.

Differentiating the constraint augmentation equations automatically expresses the acceleration constraints:

$$0 = \mathcal{J}^C \dot{u} + \dot{\mathcal{J}}^C u$$

The derivatives of the constraint augmentation equations can also be determined from the partial velocity derivatives:

$$\begin{aligned} \dot{\jmath}_{1r} &= \dot{\mathbf{C}}_r \cdot \mathbf{x} \\ &\vdots \end{aligned}$$

where the derivatives, taken in a Newtonian reference frame, of the constraint partial velocities are

$$\begin{aligned} \dot{\mathbf{C}}_r &\triangleq \frac{\mathrm{d}}{\mathrm{dt}} \mathbf{C}_r \\ &= \dot{\mathbf{v}}_r^{\text{endpoint}_1} - \dot{\mathbf{v}}_r^{\text{endpoint}_2} \end{aligned}$$

This completes the formal description of the Jacobian elements for closed chain constraints. Note that angular velocity constraints can be treated in an identical manner.

[3]Although the constraint velocity is zero, the individual *constraint partial velocities* are non-zero.

4.5 Joint Acceleration Solution

The full system Jacobian $\mathcal{J}^S$ can now be constructed using the following components: A regular Jacobian which relates the linear and angular velocities of the manipulator endpoint(s) to the the system's generalized speeds. Next augmentation equations describing the rates of change of system momenta are added. Finally, augmentation equations which ensure that the chain closure constraint is met are added. This process results in a full rank Jacobian that looks like:

$$\mathcal{J}^S = \begin{bmatrix} \mathbf{J} \\ \mathcal{J}^L \\ \mathcal{J}^H \\ \mathcal{J}^C \end{bmatrix}$$

A corresponding set of control objectives can be formulated:

$$\mathbf{a}^S = \begin{bmatrix} \mathbf{a}^{\text{endpoint}} \\ \sum \mathbf{F}^{\text{ext}} \\ \sum \mathbf{T}^{\text{ext}} + \sum (\mathbf{r}^{ext} - \mathbf{r}^*) \times \mathbf{F}^{\text{ext}} \\ 0 \end{bmatrix}$$

Relating these two quantities is the equation:

$$\mathbf{a}^S = \mathcal{J}^S u$$

from which we can solve for the derivatives of the generalized speeds corresponding to this set of control objectives:

$$\dot{u} \;=\; \mathcal{J}^{S^{-1}}(-\dot{\mathcal{J}}^S u + \mathbf{a}^S)$$

The resulting derivatives of the generalized speeds can then be used in an inverse dynamics routine to obtain corresponding joint control torques.

4.6 Implementation

The kinematics, momentum and constraint algorithms for the 2D case have been coded into C programs. The Jacobian was then formulated from these terms and used to determine desired joint accelerations given cartesian acceleration specifications. These joint accelerations, combined with the system's mass properties were used to solve for joint torques. The system's mass properties are stored in a configuration file, much like that for SDEXACT[10], and is read in at control system initialization. Changes in the dynamical system do not require recoding or recompilation of the algorithms: only the configuration file need be changed. This is true for changing mass and geometric properties, and also for the addition and removal of links, and the closure of kinematic chains.

4.7 Experimental Results

The algorithms were implemented and run on our laboratory's two-armed satellite simulator robot. Results indicated that it is possible to achieve fairly robust control of arm joint angles and/or grasped body position even in the face of poorly calibrated sensors. Calibration software and data retrieval software were unfortunately not ready too record these results in numerical form. The lack of an endpoint sensor also made it impossible to specify desired actions in inertial space: all control to date has been done using either desired endpoint positions relative to the robot base body, or desired grasped object position and orientation relative to the robot base body.

4.8 Progress Summary

Initial experimental results have been obtained which demonstrate the viability of the proposed kinematics, dynamics and control algorithms for performing control of endpoint and grasped body position. This extended computed torque control technique ran at 90Hz on our real-time computer system. The desired sample rate is on the order of 30Hz, which coincides with our vision sensor information. Therefore, the computer has three times more power than is sufficient power to perform the control. The control algorithms and videotape footage of a functional control system on our laboratory test robot were presented at the 1989 JPL Conference on Space TeleRobotics in Pasadena, California [9]. Inertial endpoint sensing is still under development, and encompases angular rate sensors, linear accelerometers, and a global position vision sensor, and a local endpoint vision sensor.

4.9 Further Research

Further experiments await the completion of our real-time software system and our vision-based endpoint sensing system. The software, ControlShell [11], will enable simple controller and simulator installation and reconfigurations under user control, and will allow display of real-time numerical data on our SUN workstations. It also facilitates the introduction of the vision based sensor systems into the control system(s). As these sensor signals become available, it becomes possible to perform demonstrations of control which are relevant to space-based robot manipulators: endpoint (tool) positioning, catching incoming objects, cooperative arm manipulation, and releasing (throwing) objects using cooperating arms.

$\sim$

Chapter 5

Experiments in Multiple Robot Cooperation

William C. Dickson

5.1 Introduction

This chapter summarizes our progress to date in the area of multiple robot cooperation. This work will eventually unite the various lines of research presently being conducted in fixed- and floating-base cooperative manipulation, and in global navigation and control of space robots. Our goal is to demonstrate multiple free-floating robots working in teams to carry out tasks too difficult or complex for a single robot to perform. Achieving this cooperative ability will involve solving specialized problems in dynamics and control, high-level path planning, and communication.

Progress Summary

Activities completed from September 1988 to February 1989 were:

- High- and low-pressure plumbing (including thruster subsystem) completed on new robot
- Computer code transfered from MATLAB to C
- Path planning algorithm completed and verified
- Server/Client architecture for multiple-robot communication programmed

5.2 Research Goals

Some of the goals of this project are:

- Cooperative manipulation and assembly by multiple robots
- Fine cooperative manipulation in presence of on-off control
- Development of control strategies for path following in presence of obstacles and large disturbances
- Path generation considering dynamic constraints and known geometric boundaries
- Task planning for complex assemblies

5.3 Experimental Hardware

The robots to be used in this research are a pair of the two-armed free-floating robots used in the LEAP experiment (see Chapter 6). The end effectors will be pneumatically driven plungers that the robots will use to cooperatively manipulate a free-floating object. The first of the two robots is near completion for use in the LEAP experiment. The main structure of the second robot has been assembled, and the high- and low-pressure plumbing is operational. The thrusters are also in place and operating.

5.4 Modelling

The robot is modelled as a three-link chain consisting of a free-floating base and a single two-link arm—resulting in the five-degree-of-freedom system described by $q_i(i{=}1,\ldots,5)$ as shown in Fig. 5.1. The set of actuators consists of eight on-off thrusters (mounted as 90° opposed pairs on each of four corners of the base), a momentum wheel on the base, and torque motors at the shoulder and elbow of the arm. For modelling and control purposes, the thrusters are grouped into two perpendicular, multi-directional, on-off-on sets. With this simplification, base control forces and torques are conveniently separated between the thrusters and momentum wheel, respectively. Thus, each robot has five controls—two thruster forces (F_1 and F_2) for two-dimensional translation, and three torques (T_1, T_2, and T_3) for orientation. f_x and f_y are manipulation forces.

The Equations Of Motion (EOM) for the five-degree-of-freedom system were derived using Kane's method [8] and can be written as:

$$\begin{aligned} \mathbf{M}\dot{\mathbf{u}} &= \mathbf{b} + \mathbf{G}\boldsymbol{\tau} + \mathbf{H}\mathbf{f} \\ \dot{\mathbf{q}} &= \mathbf{u} \end{aligned} \tag{5.1}$$

where $\mathbf{q}$, $\mathbf{u}$, $\boldsymbol{\tau}$, and $\mathbf{f}$ are defined as:

$$\begin{aligned} \mathbf{q} &\triangleq [\, q_1\, q_2\, q_3\, q_4\, q_5 \,]^T, \\ \mathbf{u} &\triangleq [\, u_1\, u_2\, u_3\, u_4\, u_5 \,]^T, \\ \boldsymbol{\tau} &\triangleq [\, F_1\, F_2\, T_1\, T_2\, T_3 \,]^T, \\ \mathbf{f} &\triangleq [\, f_x\, f_y \,]^T. \end{aligned}$$

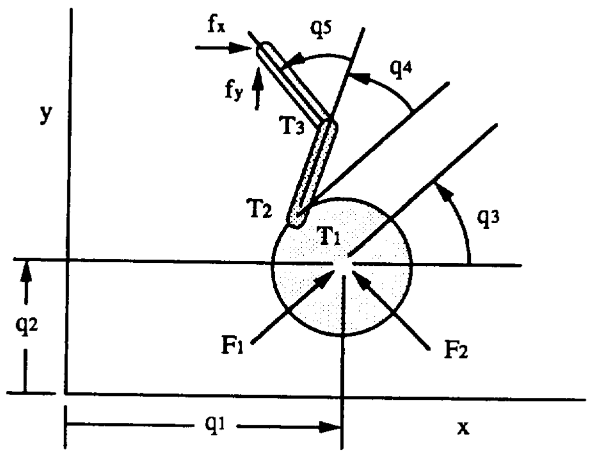

Figure 5.1: Modelling of single-arm robot

5.5 Controller

Computed torque controllers solve systems' EOM for the forces and torques required to produce desired accelerations. With the present system, given by the EOM in (5.1), G is always invertible so that the τ necessary to produce the desired acceleration vector $\dot{\mathbf{u}}_{des}$ can be given as:

$$\tau = \mathbf{G}^{-1}(\mathbf{M}\dot{\mathbf{u}}_{des} - \mathbf{b} - \mathbf{Hf}).$$

However, in our case, the first two elements of τ, F_1 and F_2, are available only in the discrete values of 0 and $\pm F_{max}$, thus not all $\dot{\mathbf{u}}_{des}$ can be produced. More precisely, only three linear combinations of the five terms in $\dot{\mathbf{u}}_{des}$ can be arbitrarily specified—the other two vary discretely due to the on-off-on values of F_1 and F_2. This problem of specifying desired accelerations was resolved as follows.

First, we define a new velocity vector $\mathbf{v}$:

$$\mathbf{v} \triangleq [\ u_1\ u_2\ u_3\ v_x\ v_y\ ]^T,$$

where v_x and v_y are the manipulator tip speeds in the x and y directions. Defining $\mathbf{a}$ as the time derivative of v,

$$\mathbf{a} \triangleq \frac{d}{dt}\mathbf{v} = [\ \dot{u}_1\ \dot{u}_2\ \dot{u}_3\ a_x\ a_y\ ]^T,$$

we find that $\mathbf{a}$ can be written as

$$\mathbf{a} = \mathbf{R}\dot{\mathbf{u}} + \mathbf{s},$$

and since $\mathbf{R}$ is invertible,

$$\dot{\mathbf{u}} = \mathbf{R}^{-1}(\mathbf{a} - \mathbf{s}).$$

We can now rewrite (5.1) in terms of $\mathbf{a}$:

$$\mathbf{MR}^{-1}(\mathbf{a} - \mathbf{s}) = \mathbf{b} + \mathbf{G}\tau + \mathbf{Hf}.$$

Using the substitutions

$$\begin{aligned} \mathbf{N} &\triangleq \mathbf{M}\mathbf{R}^{-1}, \\ \mathbf{c} &\triangleq \mathbf{b} + \mathbf{N}\mathbf{s}, \end{aligned}$$

we arrive at a new set of EOM expressed in terms of **a**:

$$\mathbf{N}\mathbf{a} = \mathbf{c} + \mathbf{G}\boldsymbol{\tau} + \mathbf{H}\mathbf{f}. \qquad (5.2)$$

The motivation for writing the EOM as in (5.2) is that the acceleration vector **a** now contains terms that directly describe the motion of the manipulated object — namely a_x and a_y, the manipulator tip accelerations in the x and y directions. Precise object control requires that arbitrary values of these accelerations be achievable. As before, only three of the five terms in **a** can be arbitrarily chosen. With two being the important tip accelerations a_x and a_y, the remaining choice is the base angle acceleration $\dot{u}_3$.

Thus, **a** can be partitioned into determined and arbitrary parts:

$$\mathbf{a} = [\, \mathbf{a}_1^T \mid \mathbf{a}_2^T \,]^T = [\, \dot{u}_1 \; \dot{u}_2 \mid \dot{u}_3 \; a_x \; a_y \,]^T.$$

$\boldsymbol{\tau}$, N, and G are similarly partitioned:

$$\boldsymbol{\tau} = [\, \boldsymbol{\tau}_1^T \mid \boldsymbol{\tau}_2^T \,]^T = [\, F_1 \; F_2 \mid T_1 \; T_2 \; T_3 \,]^T,$$

$$\begin{aligned} \mathbf{N} &= [\, \mathbf{N}_1 \mid \mathbf{N}_2 \,], \\ \mathbf{G} &= [\, \mathbf{G}_1 \mid \mathbf{G}_2 \,]. \end{aligned}$$

We can recombine the equations in (5.2) according to these partitionings to arrive at:

$$[\, \mathbf{N}_1 \mid -\mathbf{G}_2 \,] \begin{bmatrix} \mathbf{a}_1 \\ - \\ \boldsymbol{\tau}_2 \end{bmatrix} = \mathbf{c} + [\, \mathbf{G}_1 \mid -\mathbf{N}_2 \,] \begin{bmatrix} \boldsymbol{\tau}_1 \\ - \\ \mathbf{a}_2 \end{bmatrix} + \mathbf{H}\mathbf{f}.$$

Defining **W** and **P** as:

$$\begin{aligned} \mathbf{W} &\triangleq [\, \mathbf{N}_1 \mid -\mathbf{G}_2 \,], \\ \mathbf{P} &\triangleq [\, \mathbf{G}_1 \mid -\mathbf{N}_2 \,], \end{aligned}$$

we arrive at the EOM expressed in a convenient form for control purposes:

$$\mathbf{W} \begin{bmatrix} \mathbf{a}_1 \\ - \\ \boldsymbol{\tau}_2 \end{bmatrix} = \mathbf{c} + \mathbf{P} \begin{bmatrix} \boldsymbol{\tau}_1 \\ - \\ \mathbf{a}_2 \end{bmatrix} + \mathbf{H}\mathbf{f}. \qquad (5.3)$$

Given the discrete thruster forces $\boldsymbol{\tau}_1$ and the desired accelerations $\mathbf{a}_2$, we can determine the resulting base accelerations $\mathbf{a}_1$ and the required control torque vector $\boldsymbol{\tau}_2$:

$$\begin{bmatrix} \mathbf{a}_1 \\ - \\ \boldsymbol{\tau}_2 \end{bmatrix} = \mathbf{W}^{-1}(\mathbf{c} + \mathbf{P} \begin{bmatrix} \boldsymbol{\tau}_1 \\ - \\ \mathbf{a}_2 \end{bmatrix} + \mathbf{H}\mathbf{f}). \qquad (5.4)$$

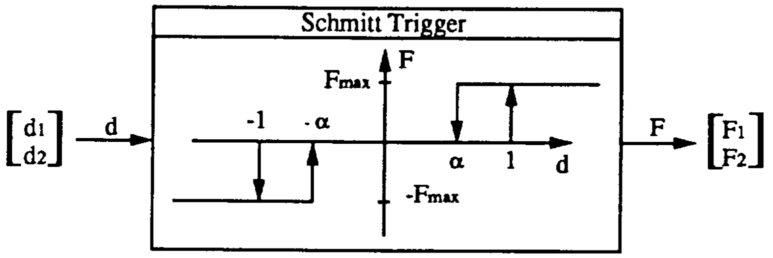

Figure 5.2: Schmitt trigger used to determine thruster forces

The control problem is thus divided between the separate tasks of first determining τ_1 (F_1 and F_2), then using these values with the desired accelerations $\mathbf{a}_2$ ($\dot{u}_3, a_x$, and a_y) to calculate the motor torques τ_2 and the resulting base accelerations $\mathbf{a}_1$.

The first candidate scheme under analysis for determining the base forces is to make F_1 and F_2 the outputs of a Schmitt trigger, as shown in Fig. 5.2. The inputs to this filter are the weighted sums of the errors in base position and velocity resolved in the two perpendicular thrust directions:

$$\begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} \cos(q_3) & \sin(q_3) \\ -\sin(q_3) & -\cos(q_3) \end{bmatrix} \begin{bmatrix} K_q & K_u & 0 & 0 \\ 0 & 0 & K_q & K_u \end{bmatrix} \begin{bmatrix} q_{1des} - q_1 \\ u_{1des} - u_1 \\ q_{2des} - q_2 \\ u_{2des} - u_2 \end{bmatrix}.$$

The parameters α, K_q, and K_u are chosen to yield desired response characteristics.

5.6 Communication Architecture

The cooperative ability of two or more robots will be limited by the rate at which vision and command data can be transfered between the robots and their coordinating agent. With this fact in mind, a server/client architecture is being developed that will allow multiple robots (clients) to access information from other robots or the coordinator through a server. High priority is placed on the rapid transfer of vision, control, and error information, while lower priority is given to information such as gas pressure and battery power levels. The server is the hub of information flow between multiple robots as well as between a robot and the coordinator, so server speed is crucial to the overall system performance.

A skeleton server/client architecture has been programmed and is operational in C. Studies are being made to determine communication protocols that maximize the speed of the server. As an example, a robot may need new vision data every sample period. In this case, at least two possible protocols can be envisioned:

In the first case, the robot must request the data at each sample period, and the server provides the information only after receiving this request. In the second case, the coordinator tells the server to provide the information to the robots at every sample period until told otherwise. In the first case, every transfer of data requires two instances of communication: one request and one actual data transfer. In the second case, once the

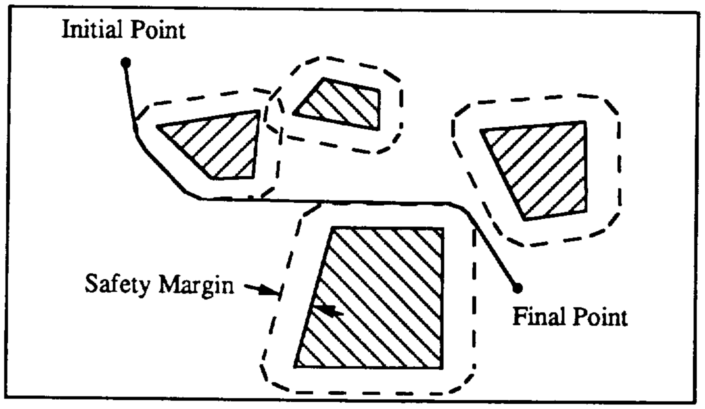

Figure 5.3: Path found by the Visibility Graph path planner

server is aware that it should send the specific data every sample period, each data transfer requires only one instance of server communication, thus the server needs to process half as many communications and its speed is increased. This type of protocol should be utilized whenever data is needed frequently. Conversely, data used infrequently should not be sent by the server unless specifically requested, lest the server would become unnecessarily burdened.

At present, the server/client architecture is being tested in the UNIX environment, which is similar to our realtime system. The architecture will be implemented on the realtime hardware as that system becomes available.

5.7 Path and Motion Planning

A path planner is required to find a viable corridor through the workspace to the desired terminal location. The path generated by the path planner will then be used to define a curvilinear coordinate system that a motion planner will use to describe the desired motion of the robots and manipulated object. In order to accommodate the possibility of unforeseen obstacles or a changing workspace geometry, the path will be frequently updated as the robots and manipulated object move toward their destination.

The visibility graph (VG) [16] is the first path-planning algorithm under study. Given a geometric representation of the workspace, the VG method searches for the shortest path from an initial to final point. Fig. 5.3 shows the path found by the VG path planner in a workspace with several obstacles that were "grown" by a safety margin to prevent collisions.

A motion planning algorithm has been developed that generates desired positions and speeds for the robots and object along a path found by the path planner. Where the path planner was concerned only with the workspace geometry, the motion planner takes into consideration performance limitations due to dynamic factors such as the robots' mass, inertia, and thrust levels. Other important factors are the robots' fields of view and the presence of unknown obstacles.

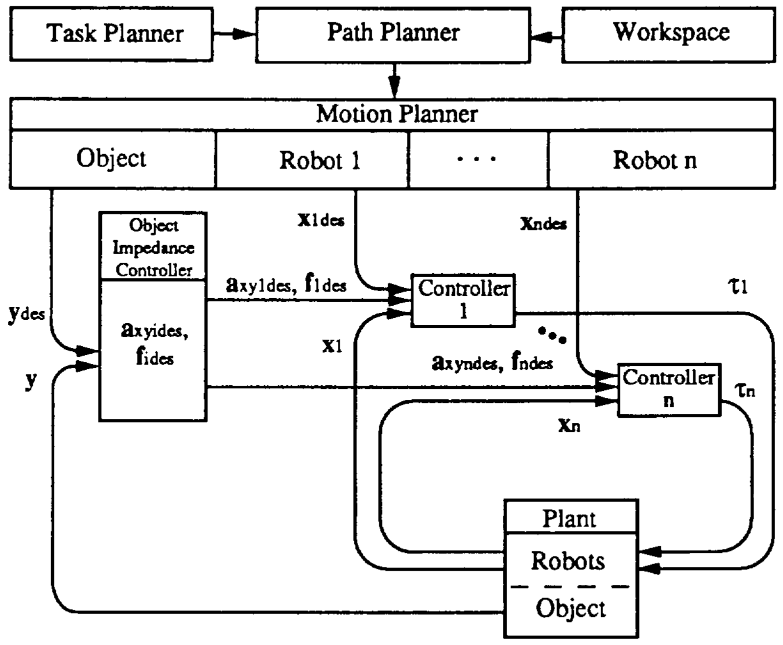

Figure 5.4: Schematic of control loop

The path-planning algorithm was recently implemented in C. The present capability of the planner is to implement a two-step path- and motion-planning procedure. First, the path planner finds the shortest path from the initial to final location using the Visibility Graph. Second, the motion planner defines path velocity limits based on the robots's limited ability to negotiate turns and decelerate (due to low thrust capability).

Fig. 5.4 shows a schematic of the control loop for the multiple-robot manipulation system.

5.8 Simulations

Simulations of a two-robot object manipulation have been successfully carried out using the matrix manipulation program Pro-Matlab [6]. One such manipulation task, shown in Fig. 5.5 , utilizes two single-armed robots transporting a beam along a straight path from A to B. An object impedance controller assigns values of a_x, a_y, f_x, and f_y to each of the two robots, then the control forces and torques are determined as discussed in Section 5.5.

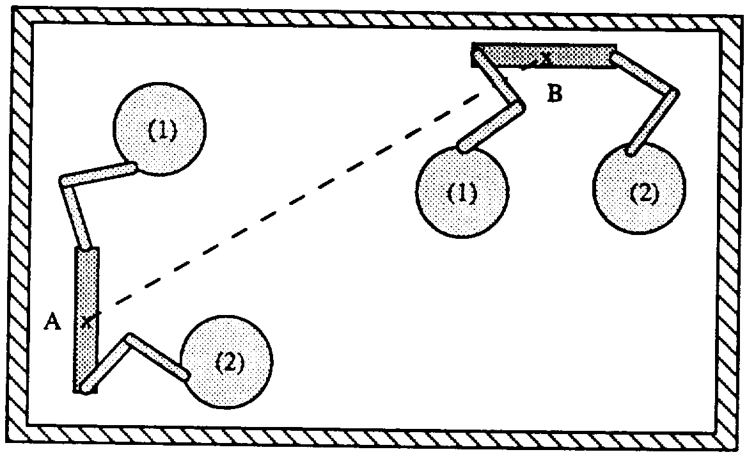

Figure 5.5: Simulation of two-robot object manipulation

5.9 Progress Summary

Advances have been made in the construction of the new robot. The high- and low-pressure plumbing has been completed and is operational. Work has begun on the digital electronics.

Simulation code that was previously written only in MATLAB has been rewritten in C.

An initial path planning algorithm has been completed and is operational.

Studies are being made to determine speed-optimizing server protocols for use with multiple-robot communication.

5.10 Future Work

The next hardware goal is to bring up the digital layer of the new robot. Other related considerations are the analog layer and the power system.

Further studies need to be made concerning the communication architecture and its implementation on the realtime system.

Studies need to be made on finding either fast or alternate means of determining the computationally intensive dynamical relationships for the complicated physical system present in this work.

Chapter 6

Experiments in Arm-Propelled Locomotion

Warren J. Jasper

6.1 Introduction

To perform assembly, maintenance and repair tasks, an autonomous robot needs to move from one workplace to another. In the case of a free-flying robot, these flights can be controlled via use of its thrusters. On orbit, however, there is a high premium on the use of propellants. Furthermore, thruster plumes may impact a target which the robot is trying to grasp, or contaminate the environment around sensitive equipment. A different approach to robot locomotion is to use its arms to push itself off from large space objects, or to traverse a space structure. This is the common mode of locomotion used by astronauts while in the Space Shuttle, and would be of equal value to a free-flying robot working in a similarly structured environment. This process, Locomotion Enhancement via Arm Pushoff (LEAP) is seen as an important technology for rendering space-based construction feasible.

Progress Summary

The major activities started or completed during the period September, 1988 through February, 1989 involved further construction of the robotic testbed. These activities were:

- Completed assembly of low pressure gas system plumbing.
- Completed fabrication of major hardware (machined) components. This includes battery packs, battery racks, analog racks, mounting plates, posts and a variety of mounting brackets. About 90% of the parts have been machined for two air cushion vehicles.
- Completed assembly/wiring of digital layer.

- Started wiring of analog electronics layer.

6.2 Research Goals

The goals of this research are to provide working demonstrations of the following advances in control theory: Non-linear control of a free-flying robot with changing degrees of freedom, Incorporation of momentum management into the control laws, and coordinated arm push-off and soft landing.

When the robot grasps an object, the dynamics change to a closed loop kinematic chain configuration, thereby introducing nonholonomic constraints. This situation, although similar to the free-flying cooperative arm manipulation experiment, has more kinematic constraints due to the apparent fixed nature of the grappled platform. Moreover, a redundant actuator, a momentum wheel, exists.

In order to minimize the need for propulsion, it is necessary to incorporate momentum management into the control laws. Momentum properties can be used to determine best times for course corrections. Thrusters will be used to effect changes in velocity, while a momentum wheel will be used for attitude control.

Tasks which are to be demonstrated are: crawling along a simulated space structure, jumping off of the structure and landing on it. The execution of these tasks, or when starting new tasks, requires switching control laws smoothly.

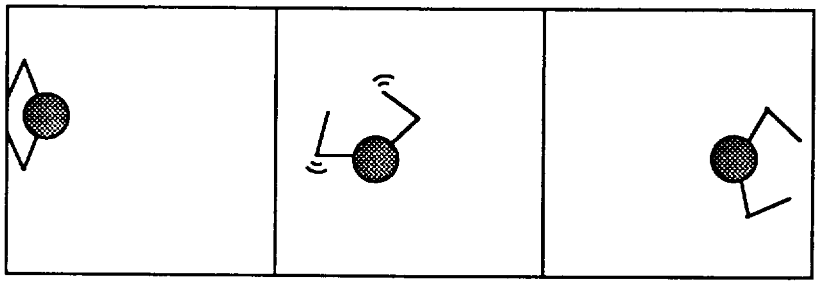

Figure 6.1: The LEAP Demonstration

6.3 The Experiment

A second generation air-cushion vehicle is being designed and built to study LEAP. This vehicle, similar to the one discussed in the previous chapters, will simulate the dynamics and activities that an autonomous space robot would perform while in the space station or maneuvering out in space. The experiment will consist of the vehicle pushing off a bar located on one side of the granite table, rotating 180 degrees, and catching itself by grasping a bar located at the other end of the table. Ideally, one would like to complete this task without the use of thrusters. However, at the point of initial release from the bar, errors in the velocity of the center of mass of the vehicle can only be corrected using thrusters.

To enhance the robustness of this approach, thrusters will be incorporated into the control laws for midcourse correction. Figure 6.1 shows the robot in three configurations: pushing off the bar, rotating, and catching itself at the other end. By incorporating crawling and leaping, the robot can position itself anywhere on the table with a minimum amount of propellant. This investigation complements current work done at the Stanford Aerospace Robotic Laboratory [3, 5] by incorporating global navigation and object manipulation into a general study of locomotion.

6.4 Fabrication and Assembly

Most of the focus of the research during the last six months was on assembling and wiring the vehicle. As mentioned in the Fourth semi-annual report [3], the overall design objective was to create a modular vehicle which consists of cylindrical layers. Each layer incorporates a major system, with five layers in all. Table 6.1 lists the systems in each layer and the fabrication/assembly status. Table 6.2 lists the status of electronics and wiring.

Task	Layer	Started	Completed
Touchdown Bar		√	√
Robotic Arms		√	√
High Pressure Plumbing	I	√	√
Low Pressure Plumbing	I	√	√
Mounting Brackets	I		
Base Plate Fabrication	I	√	√
Thruster System	II	√	√
Mounting Brackets	II		
Momentum Wheel System	II	√	√
Battery Packs	III	√	√
Battery Rack	III	√	√
Analog Rack	III	√	√
Digital Euro Card Cage	IV	√	√
Vision System	V		

Table 6.1: Fabrication Completed

I/O Interface Modules These three VME boards for interfacing the control computer to the sensor/actuator electronics have been ordered.

- Xycom XVME 290/1 Digital I/O Module: This board has 32 digital I/O lines that will be used for controlling the thruster and gripper solenoids, reading the LED signals from the hands, and enabling or disabling the batteries and safety cutout circuits.
- Xycom XVME 590/3 Analog Input Module: This board supports 16 differential channels of 12 bit analog input with a acquisition time of $25\mu s$. It will be used to read

Task	Layer	Started	Completed
Robotic Arms Wiring		√	
Pressure Sensor Wiring	I		
INS Sensor Wiring	I		
Momentum Wheel Wiring	II	√	
Thruster Wiring	II	√	
PCU Board	III	√	
RVDT Board	III	√	√
Battery Board	III	√	
Safety Board	III		
INS Sensor Board	III	√	
Force Sensor Board	III		
Motor Driver Board	III	√	√
MVME 147 CPU Board	III	√	√
XVME 590/3 A/D Board	III	√	√
AVME 9010 D/A Board	III	√	√
XVME 290 Digital IO Board	III	√	√
Transition I/O Board	III	√	√
Digital Euro Card Cage	IV	√	√
Vision System	V		

Table 6.2: Electronics and Wiring Completed

position and velocity information from our RVDT's, as well as inputs from the on-board inertial navigation unit and force sensors.

- Acromag AVME 9210 Analog Output Module. This board is equipped with eight output channels each of which is controlled by a 12 bit DAC. It will be used to issue torque commands to the five motors, as well as calibration signals to the INS unit.

6.5 Future Work

The satellite robot simulator vehicles should be near completion in the next five months. The major tasks yet to be completed are wiring, system integration and calibration. The major milestone will be to get power to the vehicle so that the onboard computer can be used to debug and calibrate the sensors and actuators. Also, development of the equations of motion and the control laws will be done.

Chapter 7

Experiments in Adaptive Control using Cooperating Arms

Roberto Ernesto Zanutta

7.1 Introduction

Recent progress in the development of adaptive control schemes for robots with cooperating arms is presented in this chapter. This work applies to fix-based robotic systems and, in conjunction with research done by other members of the ARL lab, can be extended to non-fixed base robots (specifically space robots).

7.1.1 Motivation

Some of the envisioned tasks of free-flying robots are to aid in the construction, maintenance and repair of space structures (e.g. the space station and satellites) while in orbit. Because of the high costs of placing mass into orbit, it is desirable to reduce the amount of propellant consumed by the free-flying robots while carrying out their tasks. Typical tasks performed by the robots will require them to transport and retrieve various objects. To minimize the amount of propellant required, the robots will have to know accurately the inertia properties of the objects they carry. Since it is not practical to specify the object mass properties each time a robot performs a task, some method of identification is necessary. This can be done through the use of adaptive control.

7.1.2 Research Goals

The goals of this project are:

- Demonstrate the identification of a grasped rigid object (i.e. the mass, inertia and center of mass location) by a fixed-base two-armed robot

- Demonstrate on-line control law modification in order to achieve improved performance when new object information is received
- Demonstrate control-law robustness during the identification process

7.2 Progress Summary

The following is a list of the developments during the past report period:

- The algorithms for simultaneous identification and adaptation mentioned in the previous reports were found to require too much computation when partial knowledge of the robot system was available. Hence they were not suitable for implementation in a real-time environment.
- An alternate approach to object identification was found which easily incorporates partial knowledge.
- A simulation for this approach was developed. The simulation successfully identified grasped objects and maintained "good" following of desired object trajectories during identification.
- Code for the real-time hardware was developed and partially tested.

7.3 Experimental Robot Hardware

The adaptation approaches are studied and developed with the simultaneous aid of software simulations and real-time hardware. The simulation is the same as previously reported for the LEAP project. The hardware used for this study is the two armed, fixed base robot detailed in previous reports by Stan Schneider [4]. The robot system consists of two fixed-base two-link arms cooperating to manipulate an object. The arms move in a 2-d plane perpendicular to gravity. The object floats on a granite table. It is grasped by a plunger at the tip of each arm which fits into a port on the object (the object has two ports, one for each arm). The robot arms are each actuated by two limited-angle torquer motors placed at the base of the arms. The arm positions are determined by sensing the shoulder and elbow orientations using RVDT sensors at the joints. The forces on the object are determined by force sensors at the arm tips. The force sensors measure force using strain gauges).

7.4 Control and Identification

The approach for control and identification mentioned in previous reports was found to be too complex in the presence of closed-loop kinematic chains. The approach presented was based on work by Slotine and Li of MIT [14] and Wen and Bayard of JPL [2]. The approaches required the complete resolution of the closed-loop kinematics and dynamics of the robot at all times when an object was grasped. The calculations required to do this are very expensive in terms of CPU time. In addition, partial knowledge of the robot would not necessarily simplify the adaptation process. As a result a different approach was developed. The approach was based on the separation of the identification process from the control algorithm.

7.4.1 Control Law

The control law used for the system is the object impedance controller presented by Stan Schneider [4]. The algorithm has the inherent advantage that no closed-loop dynamics have to be calculated for control. This greatly reduces the computational requirements for the controller. In addition, it is easy to incorporate new information as it is acquired by the identifier.

7.4.2 Identification Process

The identification is carried out by comparing a model of the object with the actual object motion. The parameters which best fit the model, given the object motion are determined using basic least squares techniques. An estimate of the forces on the object and the velocities of a point on the object are necessary.

The object manipulated by the arms can be described by the equations of motion:

$$m\ddot{x} + mr_y\ddot{\theta} + mr_x\dot{\theta}^2 = f_{1x} + f_{2x} \tag{7.1}$$
$$m\ddot{y} - mr_x\ddot{\theta} + mr_y\dot{\theta}^2 = f_{1y} + f_{2y} \tag{7.2}$$
$$J\ddot{\theta} = p_{1x}f_{1y} - p_{1y}f_{1x} + p_{2x}f_{2y} - p_{2y}f_{2x} \tag{7.3}$$

The first two equations describe the linear motion and the last equation describes the angular motion. The terms in the equations are:

m	$\triangleq$	mass of the object
x, y	$\triangleq$	global x and y coordinates of the object
θ	$\triangleq$	object orientation with respect to a fixed frame
r_x, r_y	$\triangleq$	distance from object center to center of mass
J	$\triangleq$	object moment of inertia
$f_{1x}, f_{1y}, f_{2x}, f_{2y}$	$\triangleq$	forces acting on the object at the ports
$p_{1x}, p_{1y}, p_{2x}, p_{2y}$	$\triangleq$	distance from ports to object center of mass

All coordinates are relative to the fixed base. The equations of motion can be expressed using four physical parameters (this is a minimum number). These can then be expressed in matrix form that can be easily implemented in a least squares algorithm. The vector form of the equations is:

$$\mathbf{f} = \mathbf{Ya} \tag{7.4}$$

where

$$\mathbf{a}^T = (m, mr_x, mr_y, mJ) \tag{7.5}$$

$$\mathbf{f(t)} = \begin{pmatrix} f_{1x} + f_{2x} \\ f_{1y} + f_{2y} \\ 0 \end{pmatrix} \tag{7.6}$$

$$\mathbf{Y(t)} = \begin{bmatrix} \ddot{x} & \dot{\theta}^2 & \ddot{\theta} & 0 \\ \ddot{y} & -\ddot{\theta} & \dot{\theta}^2 & 0 \\ \frac{1}{2}[(f_{1y} - f_{2y})d_x + (f_{2x} - f_{1x})d_y] & -(f_{1y} + f_{2y}) & f_{1x} + f_{2x} & \ddot{\theta} \end{bmatrix} \tag{7.7}$$

d_x and d_y are the distances between the grip ports on the object in the global x and y directions.

Using equations 7.4 to 7.7 the mass parameters of the object can be determined on-line. To compute these parameters measurements of the gripper forces acting on the object and the acceleration and velocity of the object are necessary. This presents a problem because acceleration measurements are notoriously difficult and suspect; therefore, some modification must be made to this identification scheme.

One way to avoid acceleration measurements or estimates is by filtering the equations of motion. Any filter with desireable characteristics can be used to do this. The one chosen for initial testing is a simple single pole low-pass filter. The equations of motion are analytically filtered by passing them through the desired filter, i.e. the equations are convolved with the filter transfer function. This can be expressed mathematically as:

$$\mathbf{z}(t) = \phi(t)\mathbf{a} \tag{7.8}$$

where

$$\mathbf{z}(t) = \int_0^t \mathbf{F(r)w(r)dr} \tag{7.9}$$

$$\phi(\mathbf{t}) = \int_0^{\mathbf{t}} \mathbf{Y(r)w(r)dr} \tag{7.10}$$

The impulse response for the low-pass filter is :

$$w(r) = \alpha e^{-\alpha(t-r)} \tag{7.11}$$

With a model of the object the parameter identification can be carried out. This is done using a recursive least-squares algorithm. Both standard least squares (SLS) and exponentially forgetting least squares (EFLS) are used.

7.5 Simulation Results

The object impedance controller with the forementioned identifier was tested out in simulation. The results showed great promise, but an ideal environment was used. No model errors, nor sensor errors were considered. This was done because it was not obvious what

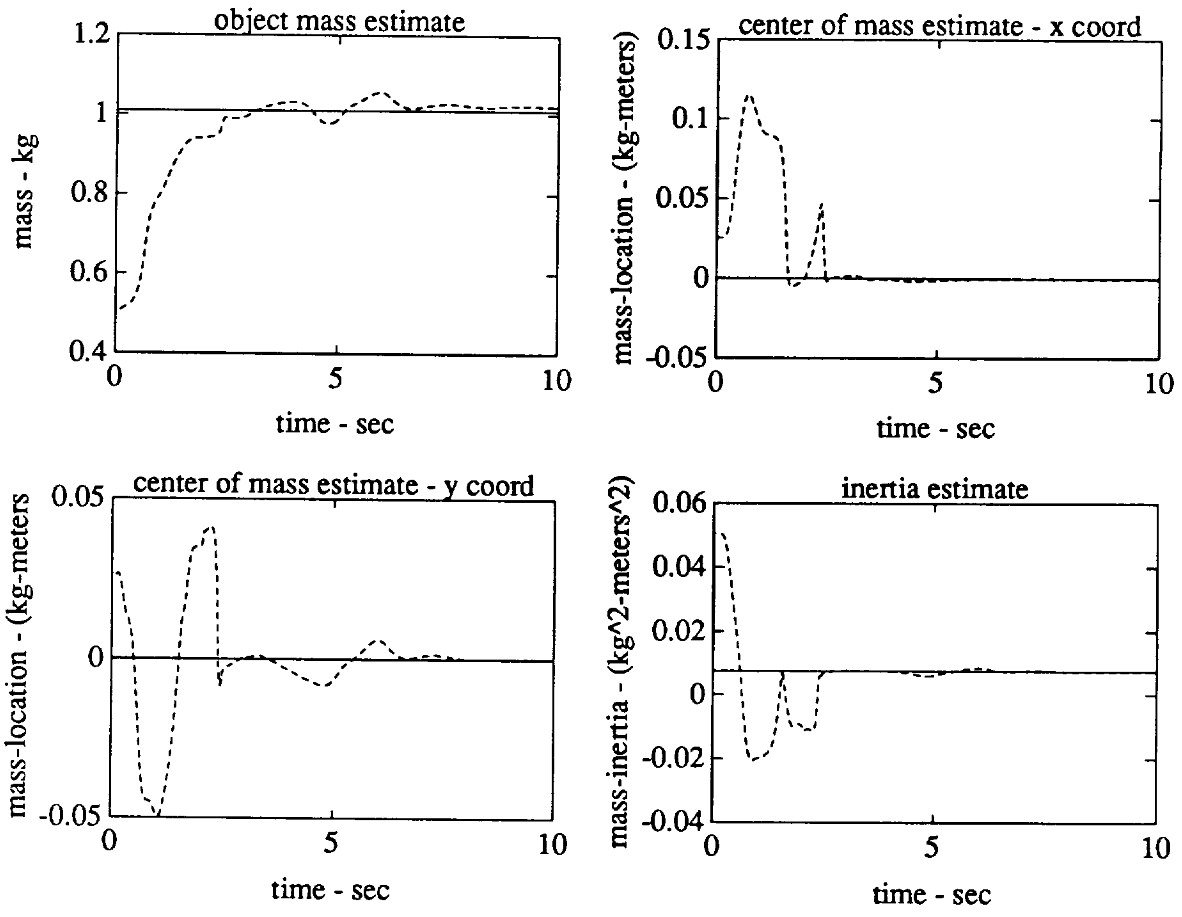

Figure 7.1: Simulation parameter estimates for a straight line slew.

their nature would be. Once the algorithm is tested on the real-time hardware the nature of the sensor and model errors can be determined. With these models the identifier can be iterated upon to yield acceptable results.

The following is an example of the simulation results. The object performed a two second straight line slew. Initially the parameters were estimated to be a = { 0.5, 0.025, 0.025, 0.05 }. The true values are a = {1.0, 0.0, 0.0, 0.0018 }. The simulation results show that the parameters were identified within ten seconds.

7.6 Future Work

The hardware for the project is fully operational. A successful working simulation of a solution to the problem has been developed. The majority of the remaining work is to successfully implement the results on the hardware. Specific tasks and issues to be completed and studied are:

- Develop the real-time code equivalent of the simulated identifier and controller.
- Evaluate the sensor subsystems (arm angle sensors and tip force sensors).

- Study and evaluate the effects of errors in quantities which are assumed to be known.
- Study the effects of object trajectories on the identification process.
- Include trajectory error information in the identification process.

Bibliography

[1] Harold L. Alexander. *Experiments in Control of Satellite Manipulators.* PhD thesis, Stanford University, Department of Electrical Engineering, Stanford, CA 94305, December 1987.

[2] David S. Bayard and John T. Wen. Simple adaptive control laws for robotic manipulators. In *Proceedings of the Fifth Yale Workshop on the Applications of Adaptive Systems Theory*, pages 244–251, 1987.

[3] Robert H. Cannon, Jr., Harold Alexander, Marc Ullman, and Ross Koningstein. NASA Semi-Annual Report on Control of Free-Flying Space Robot Manipulator Systems. Semi-Annual Report 4, Stanford University Aerospace Robotics Laboratory, Stanford, CA 94305, February 1987.

[4] Robert H. Cannon, Jr., Marc Ullman, Ross Koningstein, Stan Schneider, Warren Jasper, Roberto Zanutta, and William Dickson. NASA Semi-Annual Report on Control of Free-Flying Space Robot Manipulator Systems. Semi-Annual Report 7, Stanford University Aerospace Robotics Laboratory, Stanford, CA 94305, August 1988.

[5] Robert H. Cannon, Jr., Marc Ullman, Ross Koningstein, Stan Schneider, Warren Jasper, and Roberto Zanutta. NASA Semi-Annual Report on Control of Free-Flying Space Robot Manipulator Systems. Semi-Annual Report 5, Stanford University Aerospace Robotics Laboratory, Stanford, CA 94305, August 1987.

[6] Cleve Moler, John Little, Steve Bangert, and Steve Kleiman. *PRO-MATLAB User's Guide for Sun Workstations.* The MathWorks, Inc., Sherborn, MA, August 1987. Version 3.2-SUN.

[7] John J. Craig. *Introduction to Robotics Mechanics and Control.* Addison-Wesley, Reading, MA, 1986.

[8] Thomas R. Kane and David A. Levinson. *Dynamics: Theory and Application.* McGraw-Hill Series in Mechanical Engineering. McGraw-Hill, New York, NY, 1985.

[9] Ross Koningstein, Marc Ullman, and Robert H. Cannon, Jr. Computed torque control of a free-flying cooperating-arm robot. In *Proceedings of the NASA Conference on Space Telerobotics*, Pasadena, CA, February 1989.

[10] Dan E. Rosenthal. Order N Formulation for Equations of Motion of Multibody Systems. In G. Man and R. Laskin, editors, *Proceedings of the Workshop on Multibody Simulation*, pages 1122–1150, Pasadena, CA, April 1988. NASA Jet Propulsion Laboratory. JPL D-5190, Volume III.

[11] S. Schneider. *Experiments in the Dynamic and Strategic Control of Cooperating Manipulators.* PhD thesis, Stanford University, Stanford, CA 94305, July 1989.

[12] S. Schneider and R. H. Cannon. Experiments in cooperative manipulation: A system perspective. In *Proceedings of the NASA Conference on Space Telerobotics*, Pasadena, CA, February 1989. NASA.

[13] S. Schneider and R. H. Cannon. Object impedance control for cooperative manipulation: Theory and experimental results. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1076–1083, Scottsdale, AZ, April 1989.

[14] Jean-Jacques E. Slotine and Weiping Li. On the adaptive control of robot manipulators. In *Proceedings of the ASME Winter Annual Meeting*, pages 51–56, Anaheim, CA, 1986.

[15] Y. Umetani and K. Yoshida. Continuous path control of space manipulators mounted on omv. *ACTA Astronautica*, 15(12):981–986, 87.

[16] E. Welzl. Constructing the Visibility Graph for n line segments in $O(n^2)$. *Information Processing Letters*, 20(4):167–171, May 1985.